ASTR 288C – Lecture 7

Monday, 19 October 2009

Data Analysis I: FTools

Introduction

FTools is a collection of utility programs used to create, examine, or modify the contents of FITS data files. Some FTools are designed to analyse data stored in FITS files. For example, FTools exist to perform photometry on point sources, determine the fluxes in spectral lines, and do timing analyses on *X*-ray light curves. There are also (somewhat) user-friendly GUI tools that allow interactive browsing of FITS files and provide a more intuitive interface for running the FTools. FTools are written and distributed by NASA's High Energy Astrophysics Science Archive Research Centre (HEASARC) and are available from http://swift.gsfc.nasa.gov/docs/software/lheasoft/download.html . The FTools package forms the core of the HEASoft software system for reducing and analyzing data in the FITS format.

Each FTools task is a separate program that performs a single simple operation. The FTools are primarily a Unix-based package, although a Windows version does exist. Because Mac OS X is built upon a unix-based operating system FTools works under Mac OS X. Most of the powerful GUI interfaces run only on Unix, although the core GUI tool, FV, is available for Unix, Windows, and Mac OS X. Scripts are available for combining several FTools to perform complex tasks. All of the core FTools programs share several common design features.

- All source code is written in ANSI Fortran or C
- All scripts are written in Perl 5 (although many will work in Perl 4). The required version of Perl is distributed with FTools. You do *not* need to install Perl separately on your system
- The GUI tools are written in Tcl/Tk. The FTools distribution contains the source for the Tcl/Tk interpreters that these tools need to run and will build them for you. You do *not* need to install Tcl or Tk separately on your system.
- All use a simple standardized subroutine interface for getting the value of program parameters (e.g., the name of the input FITS file).
- All data I/O is restricted to FITS files via the FITSIO subroutine interface (or to simple ASCII format in certain cases).
- These design decisions have resulted in a software package that is exceptionally portable and can be integrated into new environments with a minimum of modification to the source code.

FTools are intended to be general purpose tools for working with FITS files. They do things like read and write FITS files, add data to existing FITS files, extract data from FITS files, and various forms of data manipulation. For example, tasks exist to perform arithmetic operations on a set of images, convolve images with various functions, and plotting data stored within FITS files. Other task have been written to manage calibration data, coordinate transformations, and spacecraft pointing data. In general an FTool is intended to perform a single task. Complex tasks are usually handled by stringing multiple FTools together.

Some FTools are designed for use with the data from specific space observatories. At present there are FTools packages that support data from the following missions.

- *ASCA*
- *Einstein*
- *EXOSAT*
- *CGRO*
- *HEAO-1*
- *INTEGRAL*
- *OSO-8*
- *ROSAT*
- *Suzaku*
- *Swift*
- *Vela*
- *XTE*

FTools have been used to construct high-level software for doing detailed scientific data analysis. The big three software packages are XSPEC, XIMAGE, and XRONOS, which work with spectral data, imaging data, and timing data respectively.

XSpec

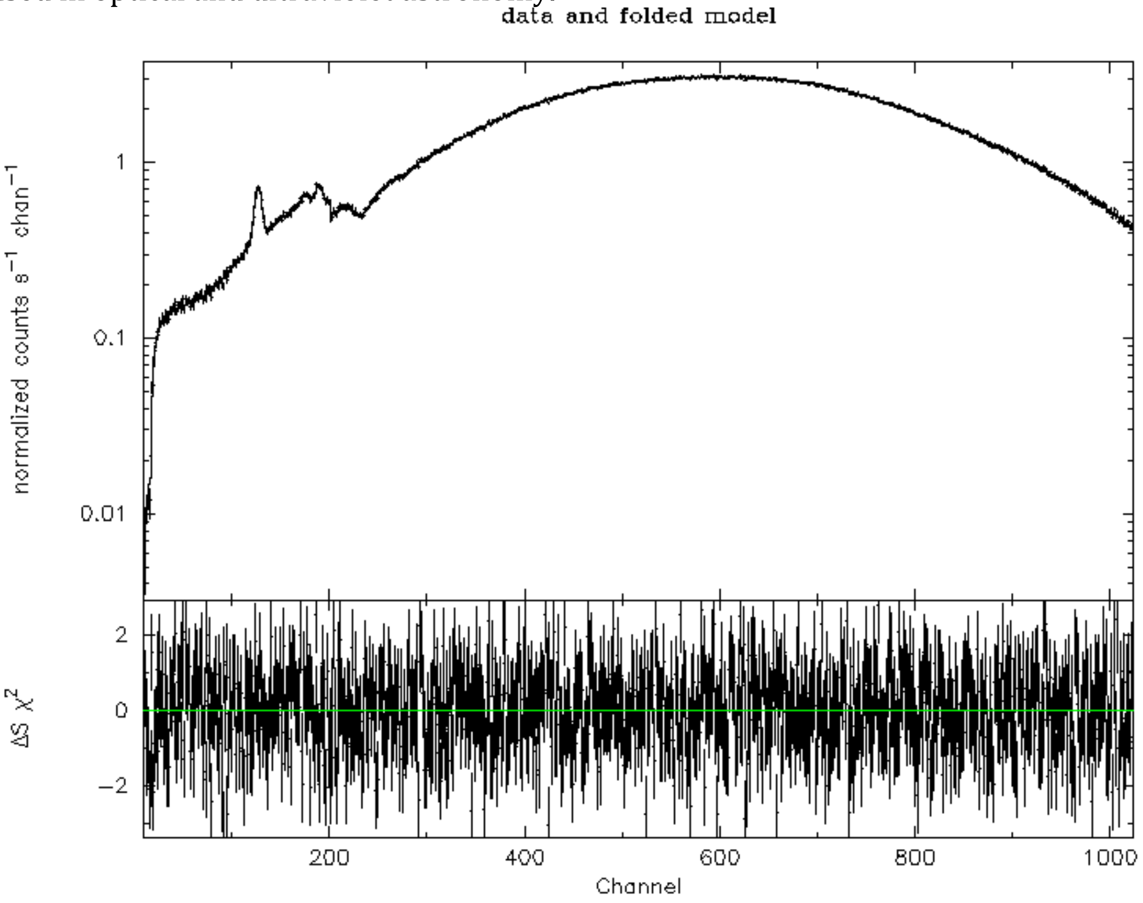
XSPEC is a command-driven, interactive, *X*-ray spectral-fitting program, designed to be completely detector-independent so that it can be used with data from any spectrometer. However, a spectrometer does not measure the true spectrum of the source. Instead it counts the incoming photons, *C*, within specific instrument channels, *I*. This observed spectrum is the convolution of the true spectrum and the instrumental response of the spectrometer. It is related to the true spectrum of the source, *f(E)* by

$$C(I) = \int f(E)R(I,E)dE$$

where *R(I,E)* is the instrumental response and is proportional to the probability that an incoming photon of energy *E* will be detected in channel *I*.

Ideally we would like to determine the actual spectrum of a source, *f(E)*, by inverting this equation, thus deriving the true spectrum, *f(E)*, for a given observed spectrum, *C(I)*. In general, however, such inversions tend to be non-unique and unstable to small changes in *C(I)*, such as noise in the data. A better method is to take a model, convolve it with the response function, and then fit the convolved model to the data using some form of parameter optimization algorithm. This is what XSPEC does.

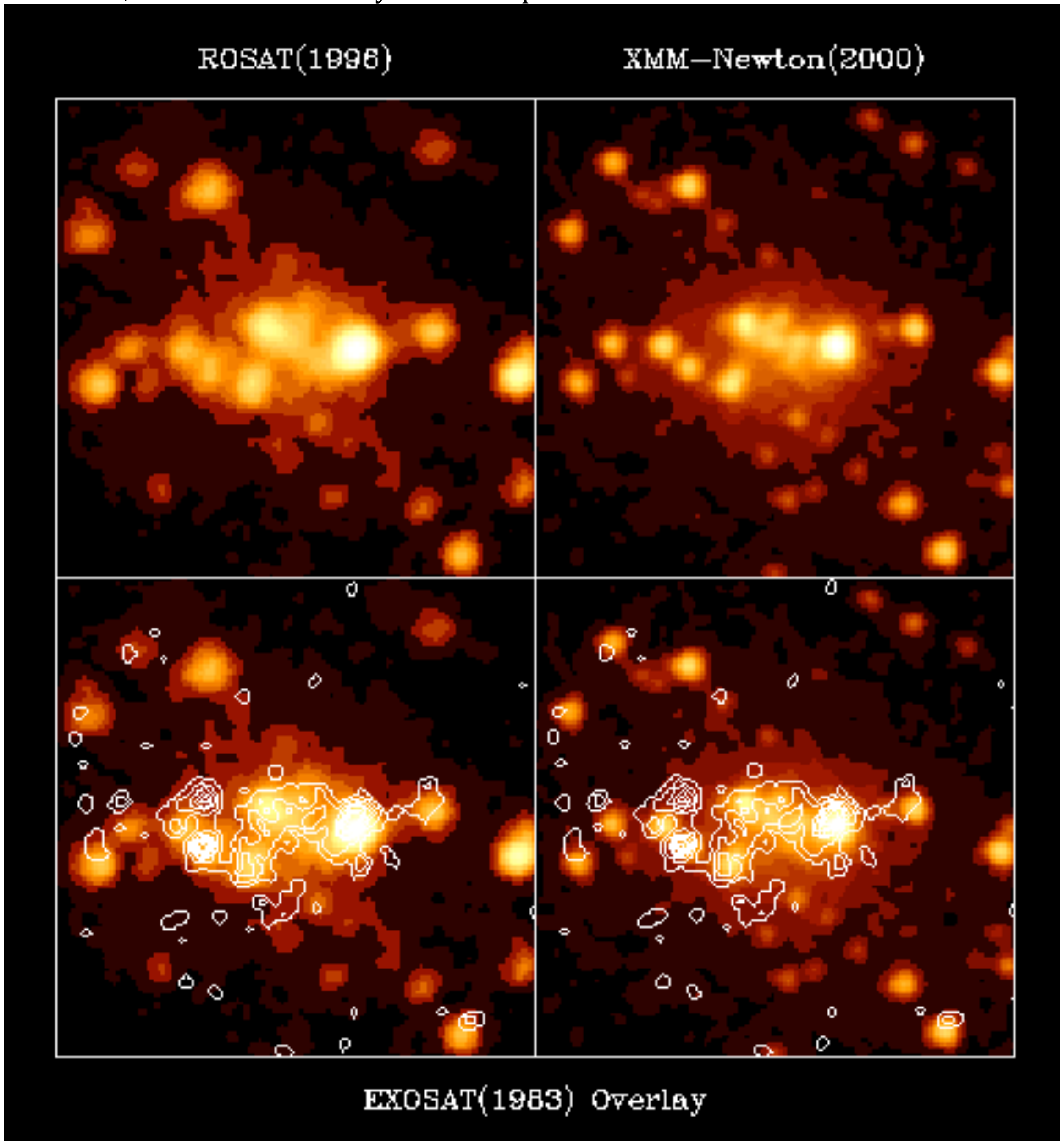
XSPEC is widely used in *X*-ray and gamma-ray astronomy, and is starting to be used in optical and ultraviolet astronomy.



This is an *X*-ray spectrum of an extragalactic source. It has been fit by a model spectrum consisting of a power law with two Gaussian emission lines. The model has been convolved with the response function of the spectrometer, which is why the two emission lines do not appear to be Gaussian. The lower plot shows the deviations of the data away from the model.

XImage

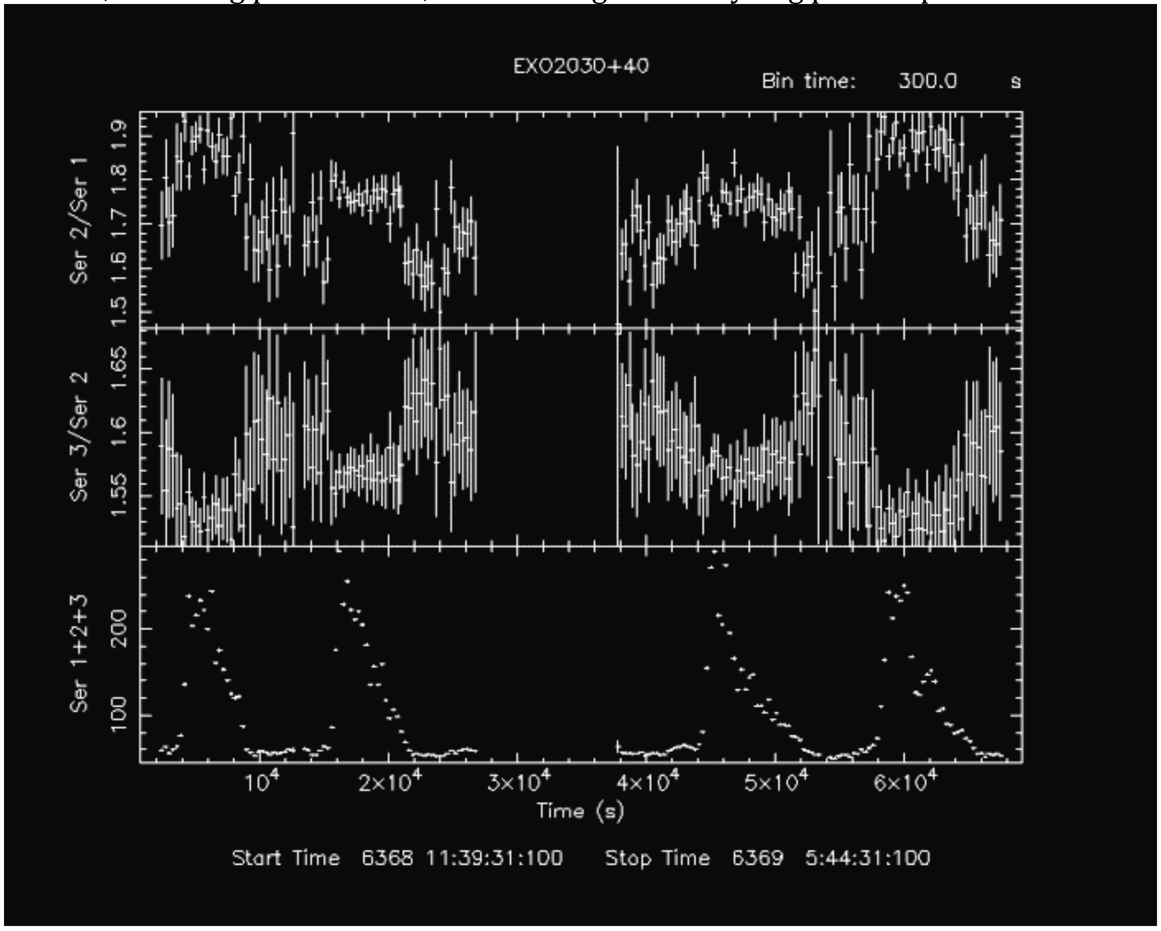
XIMAGE is a package for displaying image data, finding sources in the image data, and measuring the observed fluxes from those sources. It was designed for *X*-ray data, but it also works for gamma-ray data. Because XIMAGE was designed for high-energy observations, where the sky background is very low (often zero) the software makes some assumptions that are not always valid for optical data. Therefore, XIMAGE is not widely used for optical data.



This is an example of using XIMAGE to overlay *X*-ray intensity contours from *EXOSAT* onto *X*-ray maps of M31 obtained with *ROSAT* (left) and *XMM–Newton* (right).

Xronos

XRONOS is a tool for doing timing analyses. It works with data where the individual photons have time tags indicating the time that they were detected. It was designed for *X*-ray astronomy, but it can work on any data where the photons have timing information. It is useful for constructing light curves of rapidly varying sources, detecting periodicities, and creating and analyzing power spectra.



This is an example of using XRONOS to compute the hardness ratio for a source. Hardness ratio is the ratio of high energy photons to low energy photons. The upper two panels show the time evolution of the hardness ratios in two spectral bands. The lower panel shows the *X*-ray light curve.

Getting Help

General help information about FTools, as well as detailed help on individual tasks, can be obtained with the FHELP task.

- ➢ fhelp ftools

This command gives a brief description of each FTool task and lists the available help files. To get detailed help on any particular FTool task type

- ➢ fhelp *taskname*

where *taskname* is the name of the task that you are interested in.

The help file for each task gives a description of what the task does and a description of all of its required and optional (hidden) parameters.
It is important to note that when running an FTools task, the user is required only to enter values for the most important parameters, such as the name of the input FITS file. However, most of the tasks also have other "hidden" parameters, which have a predefined value unless the user explicitly resets them to a new value. These hidden parameters can alter significantly the operation of the task, so one must be aware of the hidden parameters available for a given task, as well as the default values of each parameter. One way to determine this information is to read the help file for the task, using the FHELP command. A quicker way to see all the task parameters and their current values is to use the PLIST command, which lists the current values of every parameter of the task. For example

- ➢ plist ftlist

gives the following output.

```
Parameters for /n/ursa/A288/sholland/pfiles/ftlist.par
   infile = 00031153009/uvot/image/sw00031153009uw1_sum.cat Input file name
   option = T                Print options: H C K I T
   (outfile = -)             Optional output file
   (clobber = No)            Overwrite existing output file?
   (include = *)             Include keywords list
   (exclude = )              Exclude keywords list
   (section = *:*)           Image section to print, eg, '2:8,1:10'
   (columns = *)             Table columns to print
   (rows = -)                Table rows or ranges to print, eg, '2,6-8'
   (vector = -)              Vector range to print, eg, '1-5'
   (separator = )            Column separator string
   (rownum = Yes)            Print row number?
   (colheader = Yes)         Print column header?
      (mode = ql)            Mode
```

Notice that only the first two parameters, "infile" and "option" are required. The rest are hidden parameters. The names of the hidden parameters are enclosed in parentheses. The parentheses are not included when running a task. Note that all tasks have a hidden MODE parameter that controls how the parameter file is used by the program, but *in general the user should not change the value of this parameter*.

The simplest way to run a task is to enter its name on the input command line followed by a carriage return, the same as any unix command. The task then will prompt the user for the value of all the required parameters. First, the current value of each parameter is displayed. If that value is acceptable, the user simply strikes a carriage return to accept it. If not, the user enters a new value followed by a carriage return. The user is *not* prompted for hidden parameters. The task simply assumes the default values. In this example all of the data in the file "source.cat" is printed.

```
➢ ftlist
  Input file name [input.fits]   source.cat
  Print options: H C K I T [T]
```

An alternate way to run a task is to list the parameter values on the command line *in the same order as listed by plist*. In the example below only the magnitude data is printed. This method *must* be used if one wants to change the hidden parameters.

```
➢ ftlist  source.cat  T  columns=MAG,MAG_ERR
```

The "columns" hidden parameter allows one to list the columns in the FITS file that one wishes to print out.

Lab Work

The goal of this lab period is to gain some basic familiarity with FTools, and to do photometry on a point source in a *Swift*/UVOT image. We will do this by using some tools to perform photometry on the optical afterglow of the gamma-ray burst GRB 081203A. GRB 081203A was a bright gamma-ray burst that was detected by *Swift* on 3 Dec 2008. The "A" indicates that it was the first gamma-ray burst to be discovered on that date. GRB 081203A had a bright optical afterglow that remained visible for more than a day. An ultraviolet spectrum of the afterglow suggests that it occurred at a redshift of $z \approx 2.1$, which corresponds to a look-back time of 10.4 Gyr.

UVOT began observing GRB 081203A starting 150 s after the initial detection. The delay was due to the time required for *Swift* to slew and point its ultraviolet/optical and *X*-ray telescopes at the burst. UVOT took observations the afterglow with all seven of its filters for several hours. We will use the exposures taken with the white filter (wavelength coverage: 1600 Å – 8000 Å) to construct a light curve for the first three hours after the burst.

First, download the following file by pointing your Web browser to http://lheawww.gsfc.nasa.gov/~sholland/astr288c/autumn_2009/index.html, then move this file to your working directory and unpack it using

```
➢ gunzip -v lecture7.tar.gz
➢ tar xvf lecture7.tar
```

You should have three files

- sw00336489000uwh_sk.img—A FITS file containing *Swift*/UVOT images of the optical afterglow of the gamma-ray burst GRB 081203A.
- src.reg—A region file for the afterglow. This tells the software where the afterglow is.
- bkg.reg—A background file for the afterglow. This tells the software what part of the image to use to estimate the background.

Now, we want to use FTools to generate a light curve for the optical afterglow. To start, use FTLIST to see how many images are in the sw00336489000uwh_sk.img file.

```
➢ ftlist sw00336489000uwh_sk.img H
```

The Primary Array contains no image data, only metadata about the images in this file. The first and second images (HDU 2 and HDU 3) are identical. This is due to technical details of how *Swift*/UVOT telemetry is converted into FITS files. You can display the individual images in this file using DS9. For example, to display the first image in the file (HDU 2) type

➢ ds9 sw00336489000uwh_sk.img\[1\] &

Notice that the first image (image 1) is stored in HDU 2. In general image *N* is stored in HDU *N+1*, so be careful when selecting images in a *Swift* file. Overlay the src.reg and bkg.reg files to see where the afterglow is and the location of the background region. To do this wait until the image is loaded. Then click the "region" button and then the "load" button. Select the region file that you want to overlay from the pop-up box. The source region file is a circular aperture with a radius of 2.5 arcsec. This was chosen to maximize the ratio between the light from the source and the light from the background in the aperture.

Decide if these regions are reasonable. Things to think about are:

1. Is the source region centred on the source?
2. Is the choice of background region reasonable?
    a. Are there sources in the background region?
    b. Is the background in the background region similar to the background that the source is sitting on?

Adjust the background regions if you think that they need to be adjusted.

The square box that is visible in the image around the afterglow (and some of the other sources in the field) is an artefact caused by coincidence loss in the UVOT's detector. For bright sources photons arrive faster than the read-out time of the detector, so some photons are lost and shifted into neighbouring pixels. This effect is automatically corrected for in the UVOT photometry software. The background region should be outside this box. The choice of background region should not be based on the background in this coincidence loss box.

To do photometry on the afterglow use the region files that were supplied (src.reg for the source region, and bkg.reg for the background region), and the UVOTMAGHIST task. UVOTMAGHIST produces a light curve by doing photometry on a single source in every exposure in a UVOT SKY image. The command to use is

➢ uvotmaghist sw00336489000uwh_sk.img white.cat white.gif CALDB CALDB CALDB src.reg bkg.reg lssfile=CALDB apercorr=CURVEOFGROWTH fwhmsig=15

Type everything on one line.

The photometry data is written to the FITS file "white.cat". A plot of the light curve is written to the GIF file white.gif. You may rename these two files to whatever you wish. The three CALDB parameters tell the task to read various calibration data from the *Swift*/UVOT calibration database.

We are leaving all of the hidden parameters at their default value *except* "lssfile", "apercorr", and "fwhmsig". "Lssfile" is set to read a large-scale sensitivity map from the calibration database. This is done to correct for small changes in the sensitivity across the UVOT detector. "Apercorr" is set to CURVEOFGROWTH because we want the photometry to be correctly calibrated. The photometry calibration assumes that the source region is a circular aperture with a radius of 5 arcseconds. If a different aperture is used the calibration will not give the correct magnitudes. To take care of this set the hidden parameter "apercorr" to "apercorr=CURVEOFGROWTH". This tells UVOTMAGHIST to use the curve of growth for a point source (read from the CalDB data) to correct the observed counts in the user-supplied aperture (in this case a circle with a radius of 2.5 arcsec) to the standard photometry aperture. The "fwhmsig" parameters tells the software what the systematic error in the aperture correction is. In our case we will assume that the systematic error is 15%.

The full set of parameters for UVOTMAGHIST (from the PLIST task, is

```
Parameters for /n/ursa/A288/sholland/pfiles/uvotmaghist.par
    infile = sw00336489000uwh_sk.img     Name of input image file
    outfile = white.cat        Name for output magnitude history
    plotfile = white.gif       Name for output magnitude history plot or NONE
    zerofile = CALDB           Name of zero points file or CALDB
    coinfile = CALDB           Name of coincidence loss file or CALDB
    psffile = CALDB            Name of PSF file or CALDB
    (syserr = no)              Include systematic errors?
    (timezero = 0)             Plot start time or 0 to determine from input data [seconds]
    (ra = 23.35)               Source RA [deg]
    (dec = -41.8231)           Source DEC [deg]
    (srcas = 3)                Source region radius [arcsec]
    (bkgas = 10)               Background region radius [arcsec]
    srcreg = src.reg           Source region file or NONE
    bkgreg = bkg.reg           Background region file or NONE
    (exclude = DEFAULT)        List of extensions to exclude or NONE
    (lssfile = NONE)           Name of LSS file or CALDB or NONE
    (frametime = DEFAULT)          Frame time [s] or DEFAULT
    (nsigma = 3.0)             N sigma for calculating faintest detection
    (apercorr = CURVEOFGROWTH)      uvotsource apercorr parameter
    (fwhmsig = −1)             uvotapercorr fwhmsig value
    (logtime = yes)            Plot time using log scale?
    (plotcol = MAG)            Column to plot
    (centroid = no)            Perform centroiding?
    (clobber = no)             Overwrite output files if they exist?
    (cleanup = yes)            Remove temporary files?
    (history = yes)            Write parameter history?
    (chatter = 3)              Chatter level
    (mode = ql)
```

For now, do not change these parameters. Many of these parameters are the same as the parameters for the UVOTSOURCE task.

Once you have performed photometry you should examine your results. Use FTLIST to extract the magnitude information from the "white.cat" file. Use this command to list the following data for each image in the SKY file. The capitalized words are the names of the data columns in the "white.cat" file.

- TIME—The time, in seconds, since the gamma-ray burst occurred.

- EXPOSURE—The effective length of the exposure, in seconds.
- MAG—The calibrated white magnitude of the afterglow.
- MAG_ERR— (the underscore between MAG and ERR is part of the column name) The one-sigma statistical error in the magnitude
- NSIGMA—The statistical significance of the source. The source is NSIGMA standard deviations above the noise in the background.

There are a few things to think about when using this data. First, most FTools report numbers to the internal precision of the calculations. This is *not* the same as the precision of the data. For example, a magnitude of 17.2541 ± 0.0253678 is not really known to a precision of four decimal places. In general, the precision of a magnitude can be estimated by looking at the location of the first significant figure of the error in the magnitude. In this example the location of the first significant figure in the error (0.0253678) is 0.02, which suggests that the magnitude is reliable only to a few hundredths of a mag. Therefore, the magnitude should be reported as 17.25 ± 0.03 mag. Use the same number of decimal places for every magnitude. In general UVOT magnitudes should not be quoted to better than two decimal places., since various systematic errors limit the accuracy of the photometry to about 0.02 mag. Similarly, UVOT times are usually rounded to the nearest second when used for photometry.

Finally, view the plot in "white.gif" using your favourite picture viewing tool. One picture viewing tool that is available under unix is XV.

```
➢ xv white.gif &
```

Save the data and your work. You will use your results from this lab in this week's homework assignment. You will also use the data from this lab in next week's lab.